Oak Ridge National Laboratory

Introduction to Lustre and NCCS Spider Parallel File Systems for XT5

Presented by:
Galen M. Shipman

April 15, 2009

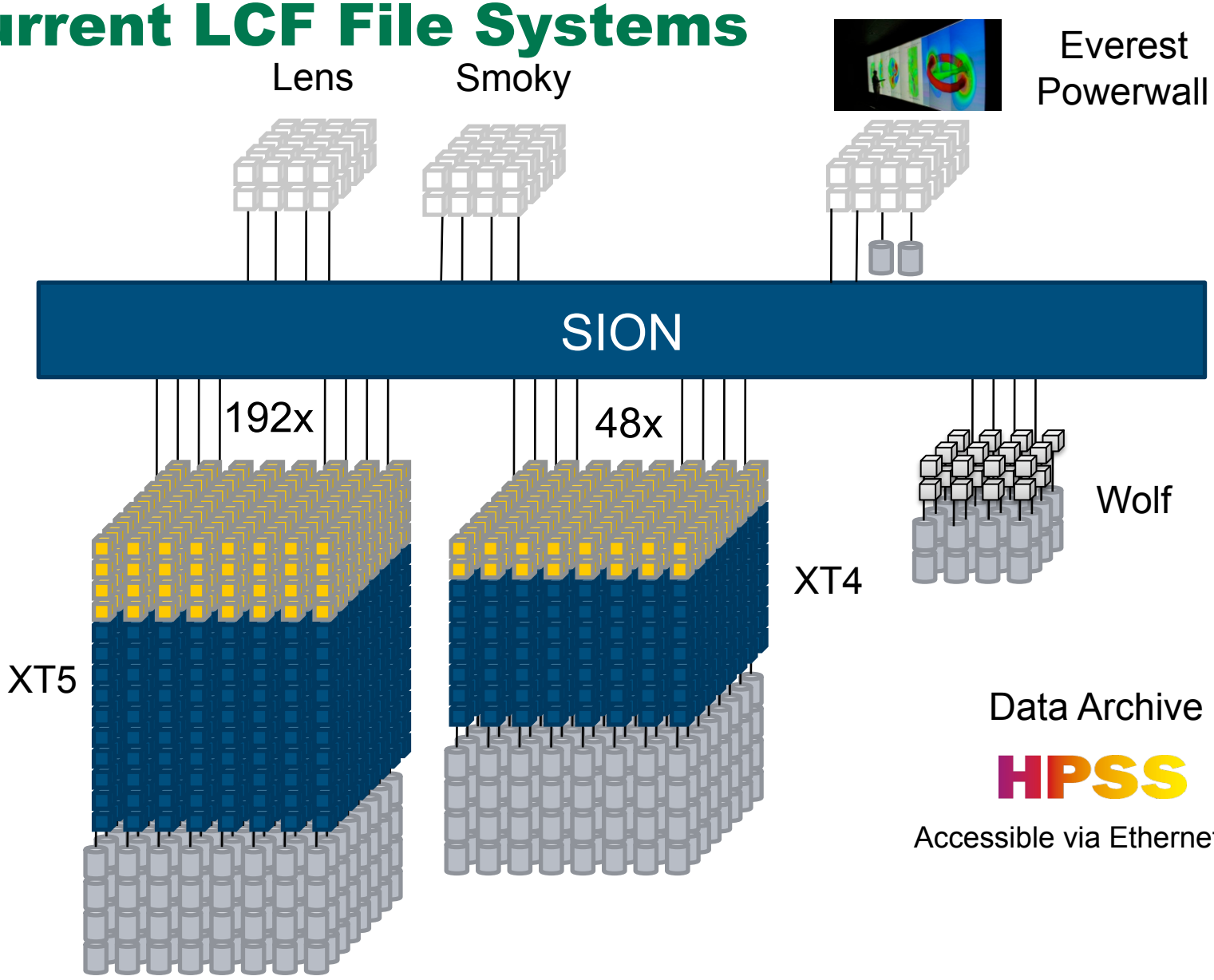
Managed by UT-Battelle for the
U. S. Department of Energy

OAK RIDGE National Laboratory

Outline

- Current file systems
- The Spider file system
- HPSS
- bbcp
- GridFTP

Current LCF File Systems



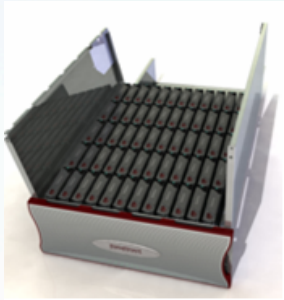
Current LCF File Systems

- XT5
  - /lustre/scratch
    - 4.2 Petabytes, > 100 GB/s, 672 OSTs
- XT4
  - /lustre/scr144
    - 284 Terabytes, > 40 GB/s, 144 OSTs
  - /lustre/scr72a
    - 142 Terabytes, > 20 GB/s, 72 OSTs
  - /lustre/scr72b
    - 142 Terabytes, > 20 GB/s, 72 OSTs
  - /lustre/wolf-ddn
    - 672 Terabytes, > 4 GB/s, 96 OSTs
    - Only available on login nodes

Current LCF File Systems …

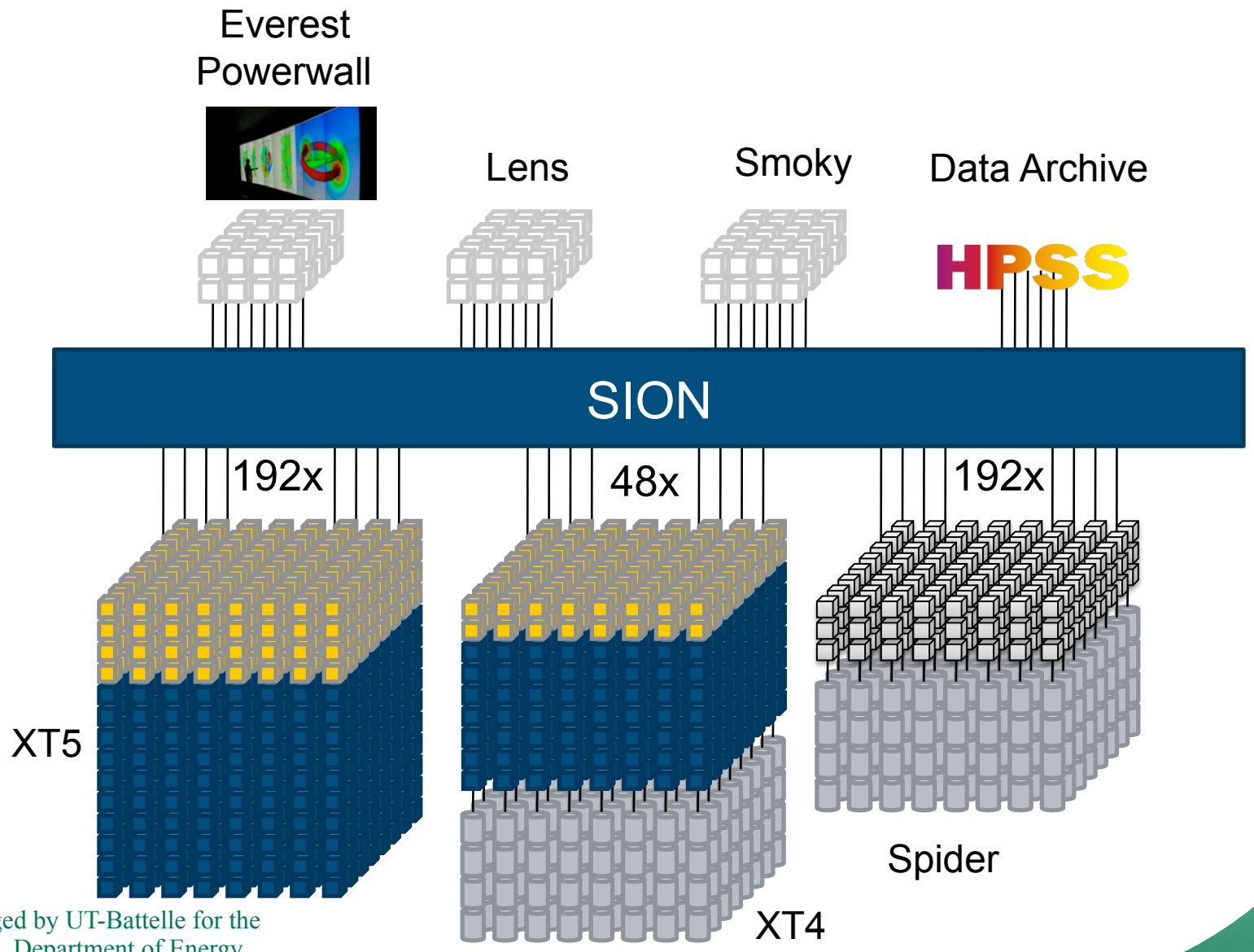
- Lens
  - /lustre/wolf-ddn
    - 672 Terabytes, > 4 GB/s, 96 OSTs
- Smoky
  - /lustre/wolf-ddn
    - 672 Terabytes, > 4 GB/s, 96 OSTs

Center-wide File System



- "Spider" will provide a shared, parallel file system for all systems
  - Based on Lustre file system
- Demonstrated bandwidth of over 200 GB/s
- Over 10 PB of RAID-6 Capacity
  - 13,440 1 TB SATA Drives
- 192 Storage servers
  - 3 TeraBytes of memory
- Available from all systems via our high-performance scalable I/O network
  - Over 3,000 InfiniBand ports
  - Over 3 miles of cables
  - Scales as storage grows
- Undergoing system checkout with deployment in summer 2009

A Center Wide High Performance File System



Benefits of Spider

- Accessible from all major LCF resources
  - Eliminates file system “islands”
  - Eliminates the need for data transfers between XT4, XT5, Lens and Smoky
    - Currently limited to Ethernet LAN bandwidth constraints
- Accessible during maintenance windows
  - Spider will remain accessible during XT4 and XT5 maintenance
  - Users will be able to access the file system from other LCF systems such as Lens and Smoky as well as remotely via GridFTP or bbcp

Benefits of Spider

- **Unswept Project Spaces**
  - **Will provide larger area than $HOME**
  - **Not backed up, use HPSS**
  - **The Data Storage council is working through formal policies now**
- **Higher performance HPSS transfers**
  - **XT Login nodes no longer the bottleneck**
  - **Other systems can be used for HPSS transfers which allow HTAR and HSI to be scheduled on computes**
- **Direct GridFTP transfers**
  - **Improved WAN data transfers**

Spider Status

- Demonstrated stability on a number of LCF systems
  - Jaguar XT5
  - Jaguar XT4
  - Smoky
  - Lens
  - All of the above..
    - Over 26,000 clients mounting the file system and performing I/O
- System Checkout is Ongoing
  - General Availability this Summer

Future LCF File Systems

- Spider File Systems
  - Accessible from XT5, XT4, Lens, and Smoky
  - /lustre/widow0
    - 4.2 Petabytes, > 100 GB/s, 672 OSTs
  - /lustre/widow1
    - 4.2 Petabytes, > 100 GB/s, 672 OSTs
  - A “wider” file system spanning all the backend storage will be reserved for special access

Achieving High Performance on LCF File Systems

- The Lustre File System provides high performance by parallelizing I/O across multiple storage arrays (hard disks) AKA Object Storage Targets (OSTs).
  - This is known as “Striping”
  - Stripe count: the number of OSTs that a file will be striped across – Default is 4
    - Lustre is limited to a maximum stripe count of 160
  - Stripe size: the amount of data placed on each OST before moving on to the next OST – Default is 1MB

Lustre File Striping

lfs setstripe <filename|dirname> <stripe size> <stripe index> <stripe count>

where

- **stripe size = the number of bytes on each OST (0 indicating default of 1 MB) specified with k, m, or g to indicate units of KB, MB, or GB, respectively**
- **stripe index = the OST index of first stripe (-1 indicating default)**
- **stripe count = the number of OSTs to stripe over (0 indicating default of 4 and -1 indicating all OSTs)**
- **For example, the command**

  lfs setstripe <dir> 0 -1 1

  sets the stripe count (width) to 1 on a directory

lfs getstripe <filename|direname>

returns the striping policy on a file or directory

Lustre File Striping

- **Querying the striping of the directory “bazz”**

```
gshipman@jaguarpf-login1:/tmp/work/gshipman> lfs getstripe bazz
OBDS:
0: fogg0-OST0000_UUID ACTIVE
1: fogg0-OST0001_UUID ACTIVE
2: fogg0-OST0002_UUID ACTIVE
3: fogg0-OST0003_UUID ACTIVE
……
……
669: fogg0-OST029d_UUID ACTIVE
670: fogg0-OST029e_UUID ACTIVE
671: fogg0-OST029f_UUID ACTIVE
bazz
(Default) stripe_count: 4 stripe_size: 1048576 stripe_offset: 0
```

Lustre File Striping

- **Setting the striping of the directory “bazz”**
  - **4 MB stripe size, start index 0, stripe count 160**

```
gshipman@jaguarpf-login1:/tmp/work/gshipman> lfs setstripe bazz 4m 0 160
gshipman@jaguarpf-login1:/tmp/work/gshipman> lfs getstripe bazz
OBDS:
0: fogg0-OST0000_UUID ACTIVE
1: fogg0-OST0001_UUID ACTIVE
2: fogg0-OST0002_UUID ACTIVE
3: fogg0-OST0003_UUID ACTIVE
….
….
669: fogg0-OST029d_UUID ACTIVE
670: fogg0-OST029e_UUID ACTIVE
671: fogg0-OST029f_UUID ACTIVE
bazz
stripe_count: 160 stripe_size: 4194304 stripe_offset: 0
```

Lustre File Striping

- **Files created in this directory inherit the striping of the directory**

```
gshipman@jaguarpf-login1:/tmp/work/gshipman/bazz> touch blah
gshipman@jaguarpf-login1:/tmp/work/gshipman/bazz> lfs getstripe blah
OBDS:
0: fogg0-OST0000_UUID ACTIVE
….
671: fogg0-OST029f_UUID ACTIVE
blah
        obdidx       objid       objid       group
             0      487196      0x76f1c           0
             1      488599      0x77497           0
             2      489369      0x77799           0
…
           157      501497      0x7a6f9           0
           158      499208      0x79e08           0
           159      497629      0x797dd           0
```

Lustre File Striping

- Jaguar XT5's local scratch currently has 672 OSTs
  - A single file cannot span all OSTs (160 max) limiting performance to ~25% of the maximum file system bandwidth
- Multiple shared files can be used to span all OSTs
    - Ensure that each shared file spans a different set of OSTs, this can be achieved using the OST index parameter in lfs setstripe

Lustre File Striping

```
gshipman@jaguarpf-login1:/tmp/work/gshipman/bazz> lfs setstripe file0 4m 0 160
gshipman@jaguarpf-login1:/tmp/work/gshipman/bazz> lfs setstripe file1 4m 160 160
gshipman@jaguarpf-login1:/tmp/work/gshipman/bazz> lfs setstripe file2 4m 320 160
gshipman@jaguarpf-login1:/tmp/work/gshipman/bazz> lfs setstripe file3 4m 480 160
```

- **4 Shared Files Spanning 640 OSTs**
- **A Larger number of shared files may be used but stripe count should be reduced to prevent any overlap of OSTs**

Lustre File Striping

- **Using lfs setstripe will not rebalance the striping on an existing file**

```
gshipman@jaguarpf-login1:/tmp/work/gshipman/bazz> lfs setstripe file0 4m 160 160
error on ioctl 0x4008669a for 'file0' (3): stripe already set
error: setstripe: create stripe file failed
```

- **Directory striping applies to files created within the directory**
- **You can reset the striping on a directory which applies to all new files created within the directory**

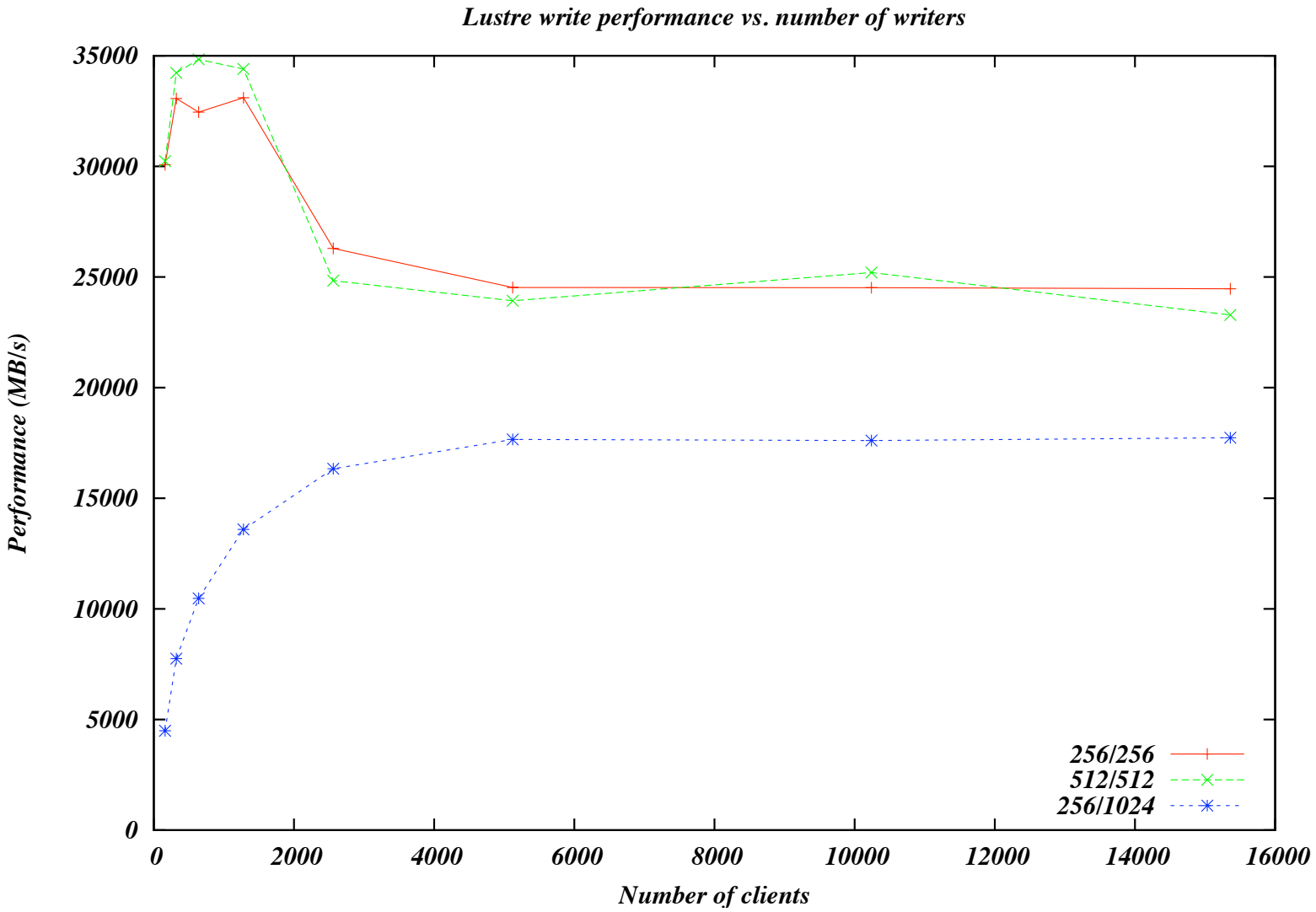
Lustre File Striping

- In addition to stripe count, stripe size is also important
  - Matching stripe size to the amount that each client writes in a given epoch will improve performance
  - As an example
    - 160 writers each writing 32 MB
    - 160 stripe count
    - 32 MB stripe size
  - Lustre locking is minimized and each writer writes to a single OST

Readers and Writers – Oh My!

- File system performance can be dramatically impacted by the number of readers/writers
  - A large number of readers/writers to a single OST can degrade performance
    - This is primarily a result of “disk thrashing/head seeks”
  - Balancing the number of readers/writers to the number of OSTs is a good rule of thumb
    - 1 – 4 readers/writers per OST with each reader/writer operating on a multiple of the stripe size

Too Many Writers Can Degrade Performance



Preventing the Thundering Herd

- 149,000 processes reading/writing to the file system will not provide optimal performance
- On a large scale simulation you can use aggregation to limit the number of readers/ writers to the file system
  - In MPI this can be accomplished by partitioning readers/writers into multiple groups and then "funneling" reads/writes to an aggregator for this group
  - Use of collective operations such as MPI_GATHER or MPI_GATHERV with the aggregator as root will improve scalability

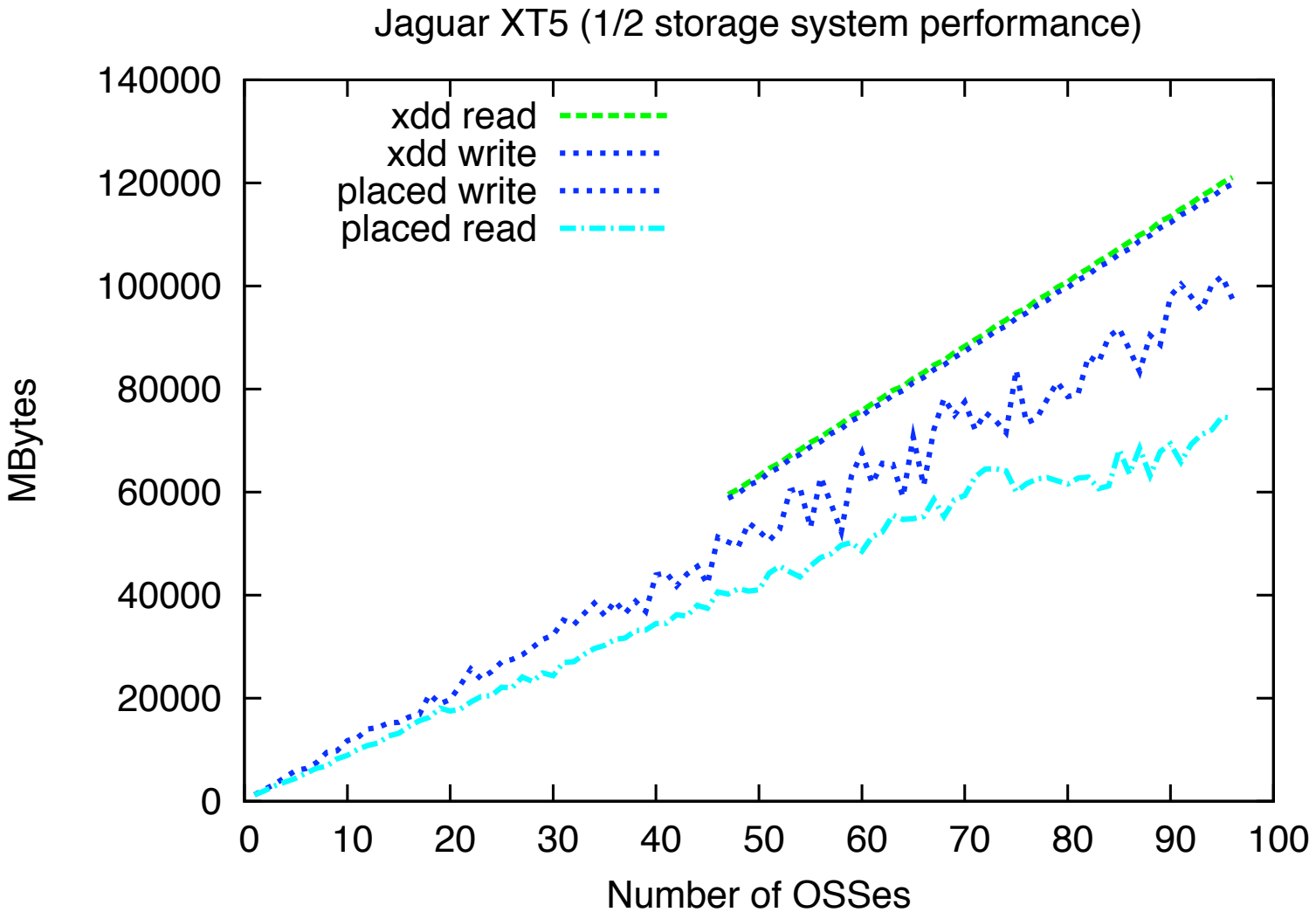
Preventing the Thundering Herd …

- MPI-IO can also be used to aggregate writes "automagically"
  - Useful for collective writes MPI_FILE_WRITE_ALL
  - Use hints to control the number of aggregators
    - export MPICH_MPIIO_HINTS="romio_cb_write=enable,cb_nodes=n"
    - Use a multiple of the number of OSTs on which the file is striped (n = 1 – 4)
  - The same applies to reads as well

Preventing the Thundering Herd …

- MPI-IO can also be used to aggregate writes
  - Useful for collective writes MPI_FILE_WRITE_ALL
  - Use hints to control the number of aggregators
    - export MPICH_MPIIO_HINTS="romio_cb_write=enable,cb_nodes=n"
    - Use a multiple of the number of OSTs on which the file is striped (n = 1 – 4)
  - The same applies to reads as well
  - If multiple files are needed (to span all OSTs) you can use separate communicators for each group, specifying the communicator in MPI_FILE_OPEN

Balancing the Number of Writers to the Number of OSTs Improves Performance



HPSS: High Performance Storage System

- A hierarchical storage system developed by a collaboration of five DOE Labs and IBM.
- ORNL's HPSS contains five PBs in over 12 million files and is growing. Installations of HPSS at other sites archive hundreds of PBs in billions of files.
- ORNL's HPSS is accessible from jaguarpf, jaguar, eugene, lens, smoky, ewok, chester, and rizzo.

Using HPSS

- **ORNL supports two interfaces to HPSS: hsi and htar**
- **hsi provides an interface similar to ftp**
  - easy to use, fine grain control (eg., chcos)
  - works well for small numbers of large files
- **htar works like tar, putting the archive in HPSS**
  - preferred interface for handling lots of small files
  - all files in a transfer treated the same

Using HPSS – hsi example

```
$ hsi
Enter PASSCODE:
hsi> pwd                          # POSIX like commands for navigating the HPSS namespace
/hpss/tpb/example
hsi> ls
/hpss/tpb/example:
file1              file2
hsi> get file2                    # transfer from HPSS to local disk
get  '/autofs/na1_home/tpb/prj/hpss/file2' : '/home/tpb/example/file2' (2003/04/09 14:09:06
   4611923 bytes, 71566.1 KBS )
hsi> put SSMHelp.7.1.0.tar        # transfer from local disk to HPSS
put  'SSMHelp.7.1.0.tar' : '/home/tpb/example/SSMHelp.7.1.0.tar' ( 10536960 bytes, 29287.4
   KBS (cos=6001))
hsi> quit
```

- **In these examples, read ~70 MB/s; write ~30 MB/s**
- **Speeds will be affected by file size and contention**

Using HPSS – htar example

```
$ htar cvf xyzzy.tar.gz w
Enter PASSCODE:
HTAR: a   w/
HTAR: a   w/hpss_mvr_tcp.wrapper
HTAR: a   w/save
HTAR: a   w/scrubble
HTAR: a   w/xddrun
HTAR: a   /tmp/HTAR_CF_CHK_13911_1239722332
HTAR Create complete for xyzzy.tar.gz. 8,704 bytes written for 4 member files, max threads:
   7 Transfer time: 0.033 seconds (0.267 MB/s)
HTAR: HTAR SUCCESSFUL
```

Using HPSS – Classes of Service

- **Selects media, copy count based on file characteristics**

COS ID	Name	Copies	File Size
5081	Disk X-Small	1,2	0 – 130K
6001	Disk Small 9840	1,2	130K – 16M
6002	Disk Medium 9840	1	16M – 530M
6003	Disk Medium 2-Copy	2	16M – 530M
6054	Disk Large_T 1-Copy	1	530M – 8G
6055	Disk Large_T 2-Copy	2	530M – 8G
6056	Disk X-Large_T	1	8G – 250T
6057	Disk X-Large_T 2-Copy	2	8G – 250T

- **Each COS represents a hierarchy with disk at the top and tape at lower levels.**

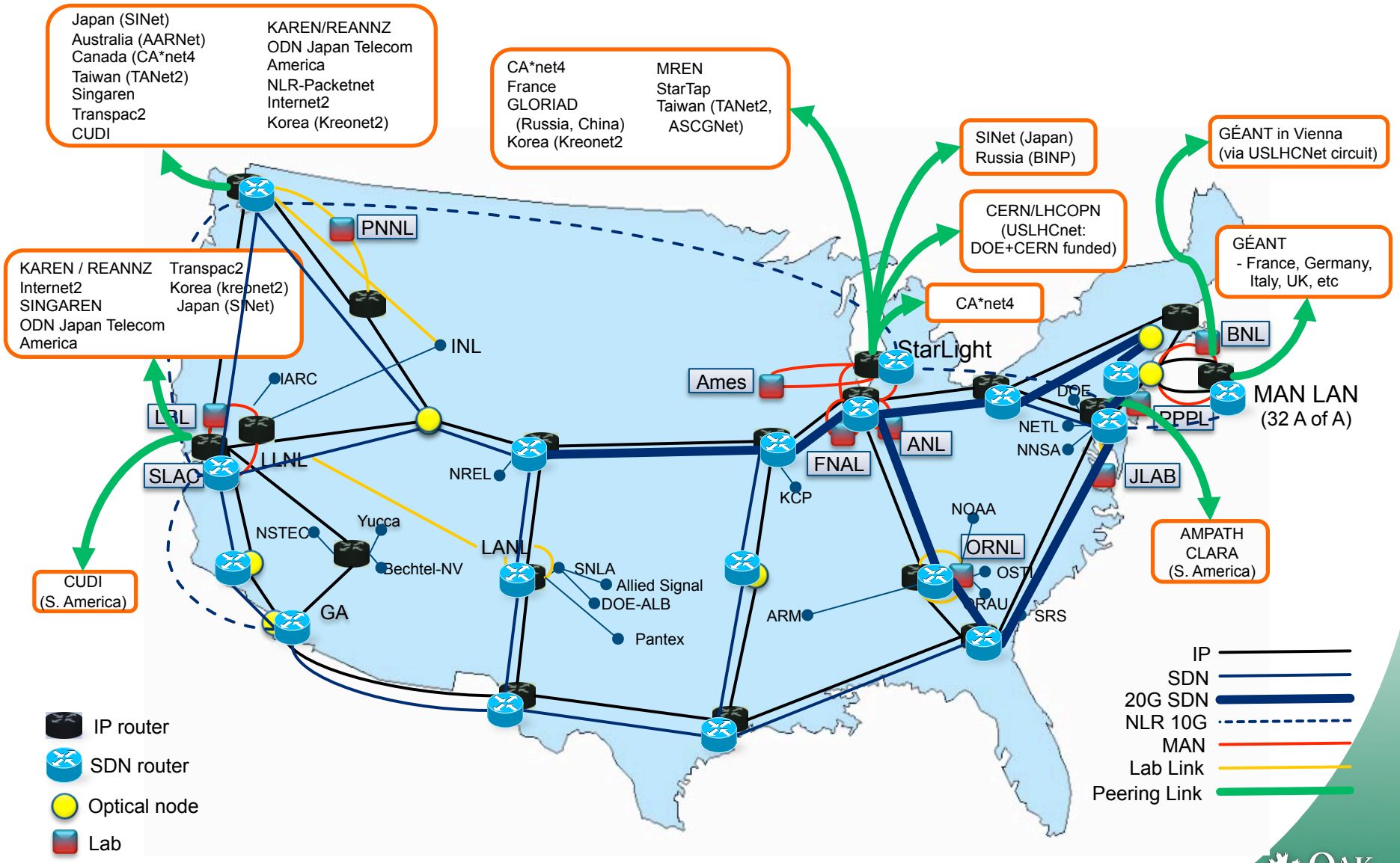
Using HPSS – Classes of Service

- But users don't have to worry about COS because we have auto-COS:
  - If file size can be determined, the file is assigned to the appropriate COS when it is created in HPSS
  - If not (eg., if the file being stored is the stdout of a running program), the file is assigned to a default COS and moved later (if necessary), once the file size is known.
- NCCS users get two copies of each file by default. Please set "copies=1" in .hsirc unless you need two copies of your files.
- When possible, it is best to avoid changing a file's COS unless necessary since it is an expensive operation.

Using HPSS – Best Practices

- Make a practice of regularly copying any important data from scratch areas into HPSS so it's not lost. (Files older than 14 days are periodically deleted from scratch areas.)
- Where possible, consider storing/retrieving data directly to HPSS from your running job (on lens, smoky, ewok; not supported on XT4/XT5). This will require passwordless (keytab-based) access to HPSS, which can be requested from User Services.
- Lean toward large files (Gigabytes). Consolidate small files using htar or by storing tar files containing smaller files.
- Be respectful on login nodes when using HSI or HTAR, i.e. don't spawn a large number of HSI or HTARs concurrently as this impacts other users
- Avoid the need for expensive change COS operations by making the file size available to hsi/htar (i.e., avoid piping data directly to hsi; rather, save the file to disk and let hsi read it from there).

Data Transfers over the WAN



BBCP

- **Provides high performance data transfers over the WAN**
  - **Files are transferred using multiple TCP/IP streams (unlike SCP which uses a single stream)**
  - **Data is not encrypted**
- **BBCP is available on Jaguar XT4 and XT5**
  - **Remote sites may require BBCP to be installed**
    - **Users can generally install bbcp in their home directory if not available on a remote system**
    - **Firewall rules may need to be modified by administrators at both the remote site and at NCCS**
- **Data transfer rates vary based on numerous factors**

Using BBCP

```
/opt/public/bin/bbcp -f -P 10 -V -s 4 -w 48M -n -T 'ssh -x -a -
oFallBackToRsh=no %I -l %U %H /usr/common/usg/bin/bbcp'
data.tar.gz blorg@franklin.nersc.gov:data.tar.gz

/usr/common/usg/bin/bbcp -f -P 10 -V -s 4 -w 48M -n -T 'ssh -x
-a -oFallBackToRsh=no %I -l %U %H /usr/common/usg/bin/bbcp'
data.tar.gz blorg@jaguarpf.ccs.ornl.gov:data.tar.gz
```

- **-s : controls the number of TCP/IP streams to use for transfer**
- **-T : command to start bbcp on remote host**
  - **Generally only the path to bbcp needs to change**
- **Read "bbcp –help" for more information**

GridFTP at NCCS

- GridFTP is a parallel file transfer utility for moving massive amounts of data
- Similar to bbcp but allows for:
  - Encrypted transfers
  - Workflow automation using grid certificates
- NCCS is deploying dedicated transfer nodes for staging data in and out of the center
- Nodes are being tuned for transfers over the wide-area to other centers, beginning with NERSC

GridFTP at NCCS

- Using these nodes, we expect >200MB/s transfers to and from the Spider file system
  - Data can be transferred directly to/from spider
- Currently in limited deployment
- For more information on using GridFTP and the transfer nodes, see http://www.nccs.gov/user-support/general-support/data-transfer/
  - These pages are still under development

Putting it All Together

- **NCCS is aggressively deploying a common backplane of services for users**
  - **The Spider Center wide file system**
  - **Higher performance HPSS transfers over our System Area Network to/from Spider**
  - **GridFTP servers for high performance WAN transfers directly to Spider**